VERİ AKIŞ DİYAGRAMI – KAVRAMSAL SINIF DİYAGRAMI

Sistem Analizi ve Tasarımı Dersi

İçindekiler

Veri Akış Diyagramları

Veri akış diyagramları

1. dışsal birim (varlık)
2. proses (işlem)
3. veri deposu
4. veri akışı

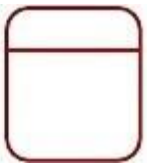
olmak üzere dört adet eleman kullanılarak çizilmektedir.

Dışsal Birim 

Veri kaynağı yada verinin varış noktaları olan sistem dışı olguları belirlemek için dikdörtgen kullanılır. Aynı varlık bir diyagramda tekrarlı olarak çizildiğinde, tekrarlı çizilen kaynağın kenarına çizgi koyarak, aynı varlığı temsil ettiği belirtilir.

Dış olgulara örnekler;

- Bir kişi, Müşteri veya Öğrenci.
- Bir şirket veya örgüt, Banka veya Tedarikçi.
- Örgüt içi bir birim, Satış departmanı.

Proses (İşlem) 

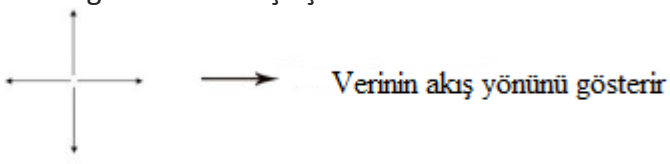
- Prosesler, yapılan bir fonksiyonu ya da işlemi tanımlar. Proseslere bir isim ve numara verilir. Proses ismi olarak emir cümleleri kullanmak uygun olacaktır (Net geliri hesapla gibi).

Veri Deposu 

- Analiz esnasında, verilerin depolanmasına ihtiyaç duyulan yerler olur. Bu yerler veri deposu olarak isimlendirilir. Veri deposu veritabanı tabloları, xml gibi veri saklama üniteleri olabilir. Her bir veri deposu için ayrıca bir de isim verilir.

Veri akışı

- Bir noktadan diğerine veri akışı için oklar kullanılır.

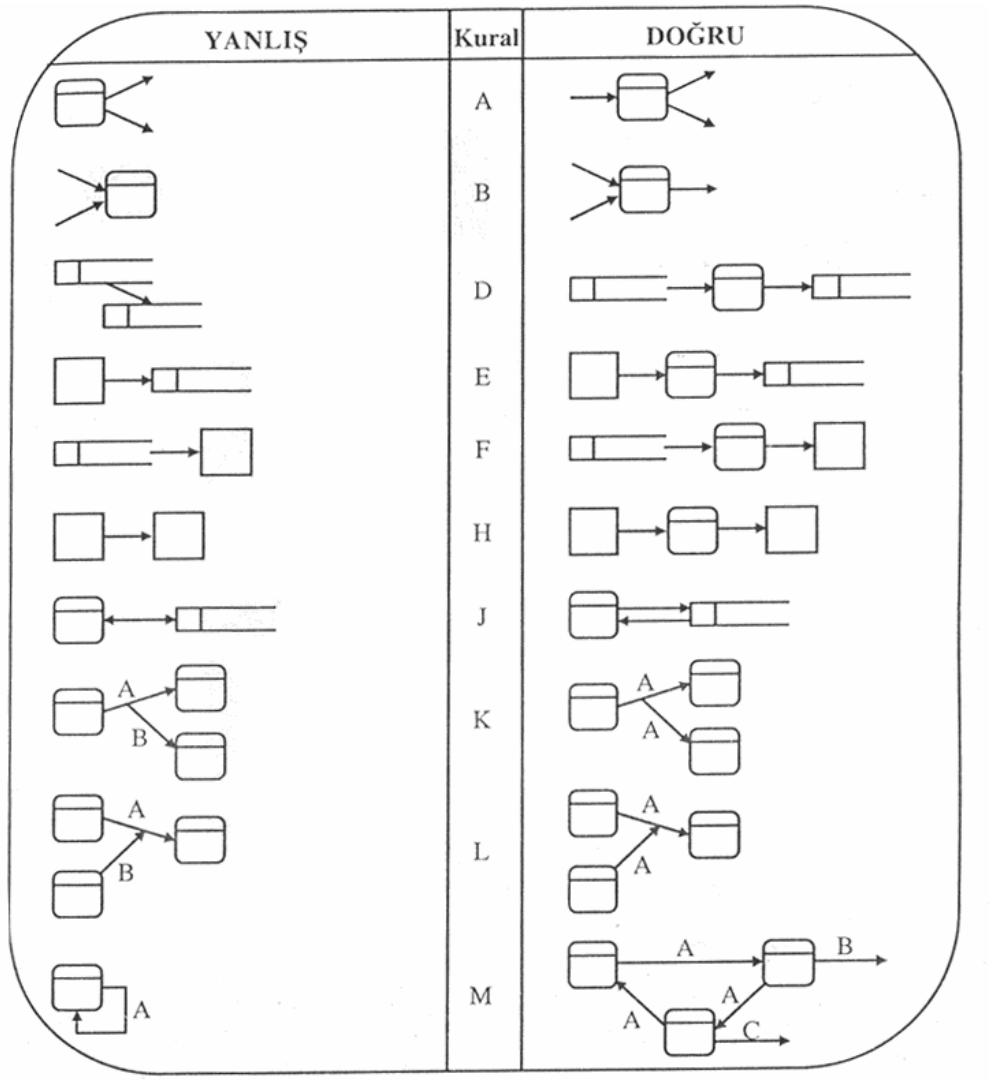


  - Sistemde hareket eden veriyi tanımlar. Veri bilgisi oka yazılarak belirtilir.
  - Ok ucu veri akış yönünü gösterir. Bir veri akışının veri deposuna gitmesinin anlamı, güncellemedir. Bir veri deposundan veri akışının çıkmasının anlamı, getirme ya da kullanmadır.

Veri Akış Diyagramları Çizim Kuralları

Veri akış diyagramları aşağıda belirtilen kurallara uygun olarak çizilmesi gerekmektedir.

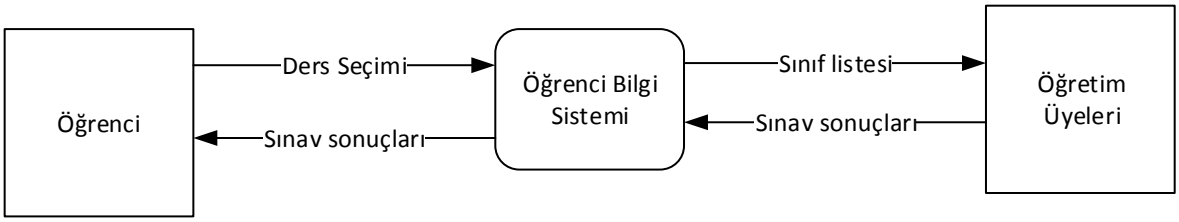
1. Hiçbir proses sadece çıktılara sahip olamaz. (Şekil 1. A)
2. Hiçbir proses sadece girdilere sahip olamaz. (Şekil 1. B)
3. Veri, bir veri deposundan diğerine doğrudan taşınamaz. Veri bir prosesle taşınmalıdır. (Şekil 1. D)
4. Veri, doğrudan bir dışsal kaynaktan bir veri deposuna taşınamaz. Dışsal birimden veriyi alan ve veri deposuna yerleştiren bir prosesle taşınmalıdır. (Şekil 1. E)
5. Veri, bir veri deposundan doğrudan bir dışsal birime taşınamaz. Veri bir prosesle taşınmalıdır. (Şekil 1. F)
6. Veri doğrudan bir varlıktan diğerine taşınamaz. (Şekil 1. H)
7. Bir veri akışı, semboller arasında tek bir akış yönüne sahip olmalıdır. Bir proses ve veri deposu arasında, veri deposundan okuma ve proseste güncellemenin gösterilmesi için her iki yönlü akış olabilir, ancak bunların iki ayrı ok şeklinde gösterilmesi gerekir. (Şekil 1.J)
8. Çatallı bir veri akışının anlamı, aynı verinin ortak bir lokasyondan iki ya da daha fazla farklı prosese, veri deposuna yada dışsal birime gitmesi demektir. (Şekil 1. K)
9. Veri akışlarının birleşmesinin anlamı, aynı verinin herhangi iki ya da daha fazla farklı prosesten, veri deposundan ya da dışsal birimden, ortak lokasyona gelmesidir. (Şekil 1. L)
10. Bir veri akışı, doğrudan aynı prosese geri dönemez. Veri akışını alıp, başka veri akışlarını üreten ve başladığı prosese orijinal veri akışını getiren en az bir prosesin olması gerekir. (Şekil 1. M)



Şekil 1. Veri akış diyagramı kuralları

Taslak (Context) Diagram

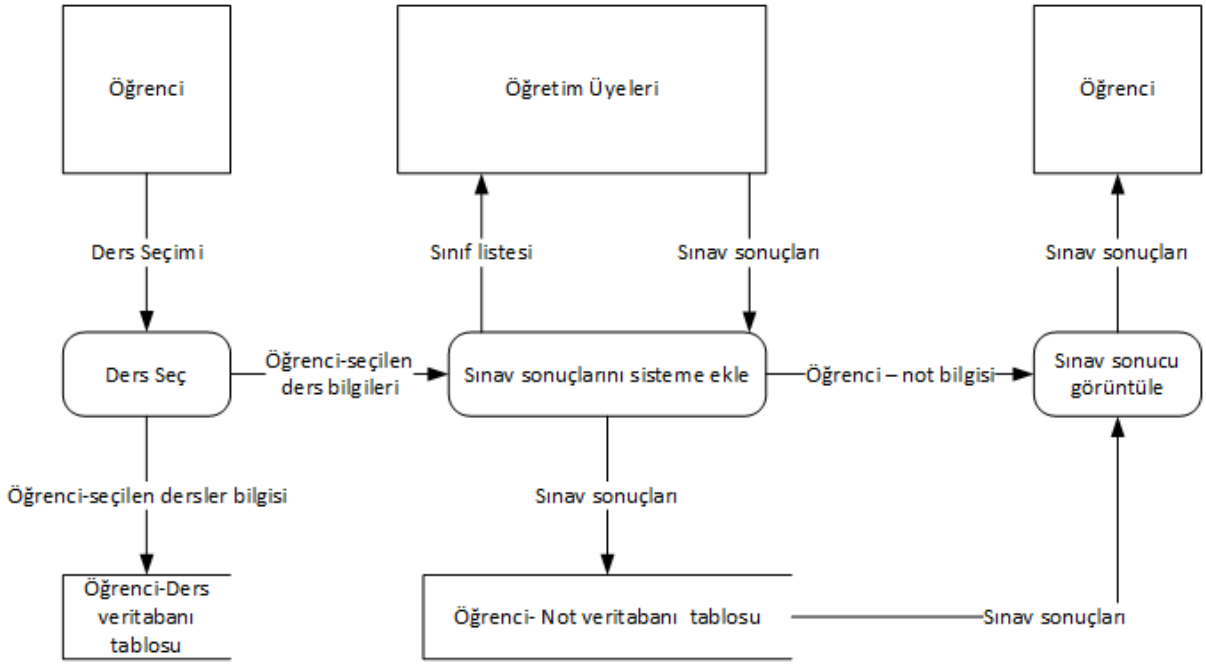
Sistemin dışsal birimlerinin sistem ile ilişkisini basitçe gösterildiği diyagramdır. Tek bir işlem olur ve sistemin ismi ile belirtilir.



Şekil 2. Örnek taslak diyagram

Birinci Seviye Diyagram

Taslak diyagramda çizilen grafiğin, bir seviye detaylandırma yapılan versiyonudur. Diyagram detaylandırma işleminde, veri akış ve diğer elemanlar, bir üst diyagram ile uyumlu olmalıdır. Örneğin, taslak diyagramda "öğrenci" varlığından sisteme doğru "ders seçimi" veri akışı var ise, alt seviyesinde de bu veri akışı olmalıdır.



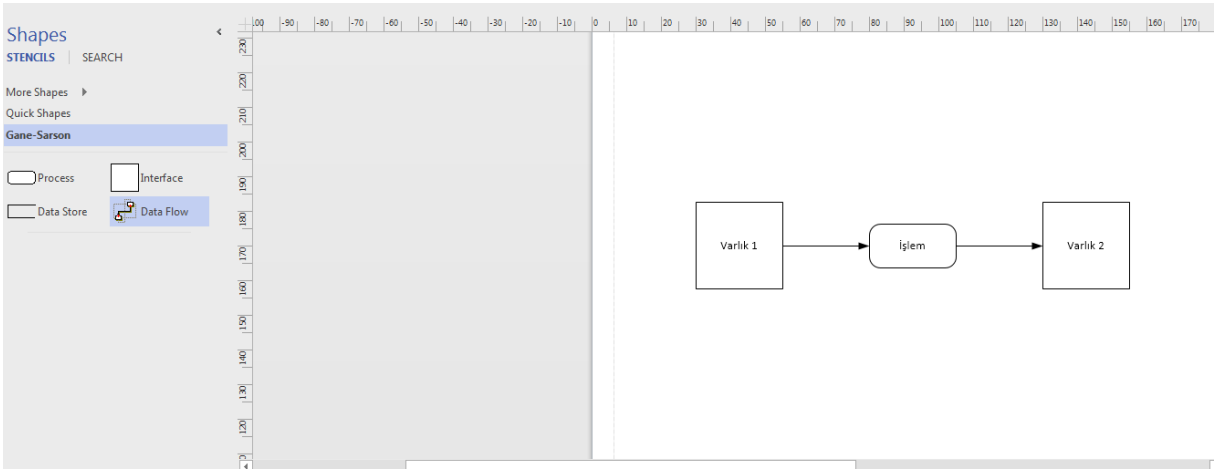
Şekil 3. Örnek birinci seviye diyagram

Veri akış diyagramı çizimi

Diyagram çizimi için Microsoft Visio programını kullanabiliriz. Visio programı çizilen diyagramın kurallara uygunluğunu denetlememektedir. MS Visio 2016'da veri akış diagramı kategorilerden "Software and Database" seçip, "Data flow model diagram" seçeneği seçilir. "Metric Units" seçilip, "create" butonuna basılır.

MS Visio 2003'de "Software" kategorisi içerisinde "Data Flow Model Diagram(Metric)" seçilir.

Veri akış diagramının dört elemanı sol taraftaki şekiller kısmında bulunmaktadır. "Varlık" elemanı "interface" olarak isimlendirilmiştir. Sürükle bırak ile çizim yapılmaktadır. Elemanların üstüne çift tıklayarak isimlendirme yapılmaktadır. Çizim sırasında ok yönüne dikkat edilmelidir.



Şekil 4. Veri akış diyagramı çizimi (MS Visio 2016)

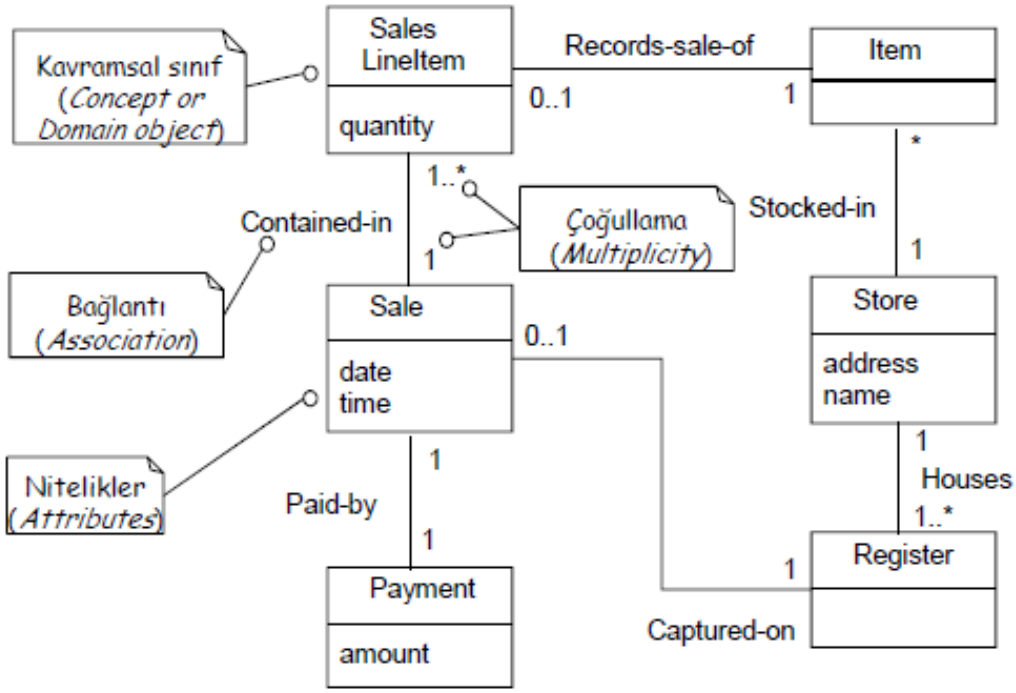
Çizim sırasında okların karışmamasına dikkat ediniz. Bunun için, aynı varlığı diyagramın diğer kısmında tekrardan yazarak, ok karmaşasının önüne geçilebilir. Bu durumda, diğer varlığın kenarına çizgi işareti konularak, aynı varlığı temsil ettiği belirtilir. MS Visio için bu özellik bulunmamaktadır, aynı isimlendirme yapmak yeterlidir.

Okları elemanlar ile eşleştirirken, veri akışı okunu, kaynak elemanın orta kısmına getirdiğinizde, elemanın üzeri yeşil olmaktadır, eşleştirmeyi bu şekilde yapınız. Böylece, veri akış oku o elemana bağlanmış olmaktadır. Elemanın yerini değiştirdiğimizde, bağlanan ok da onunla birlikte hareket eder. Kaynak elemana bağladıktan sonra, okun uç kısmından tutup uzatarak, hedef varlığın orta kısmına getirdiğimizde- elemanın üstü yeşil olacaktır – oku bırakarak, hedef varlığa bağlanır. Bu sayede elemanlar yer değiştirdiği zaman, bağlı olan veri akış okları da onlar ile birlikte hareket eder.

Ek olarak, ok karmaşasını önlemek için, fazla sayıda girdi-çıktıya sahip elemanları daha uzun çizerek, okları birbirinden uzaklaştırabiliriz. Elemanlara bağlı olan oklara tıklayıp, fare imlecini okun orta kısmında nokta ile belirtilen eklem yerine getirdiğimizde, fare imlecinde sağa sola hareket ettirebileceğimizi gösteren işaret çıkacaktır. Bu kısımdan sürükleyerek, elemanlara bağlantısını bozmadan, oku kaydırabiliriz.

Kavramsal Sınıf Diyagramı

Sistemde kullanılacak sınıflar, sınıf üyeleri ve sınıflar arası ilişkilerin tanımı ile oluşturulan diyagramdır. Kavramsal sınıf diyagramlarında, sistemin iç tasarımın ayrıntılarını göstermeden, gereksinimleri karşılayacak sınıfları, sınıfların niteliklerini ve sınıflar arası bağlantıları gösterir.

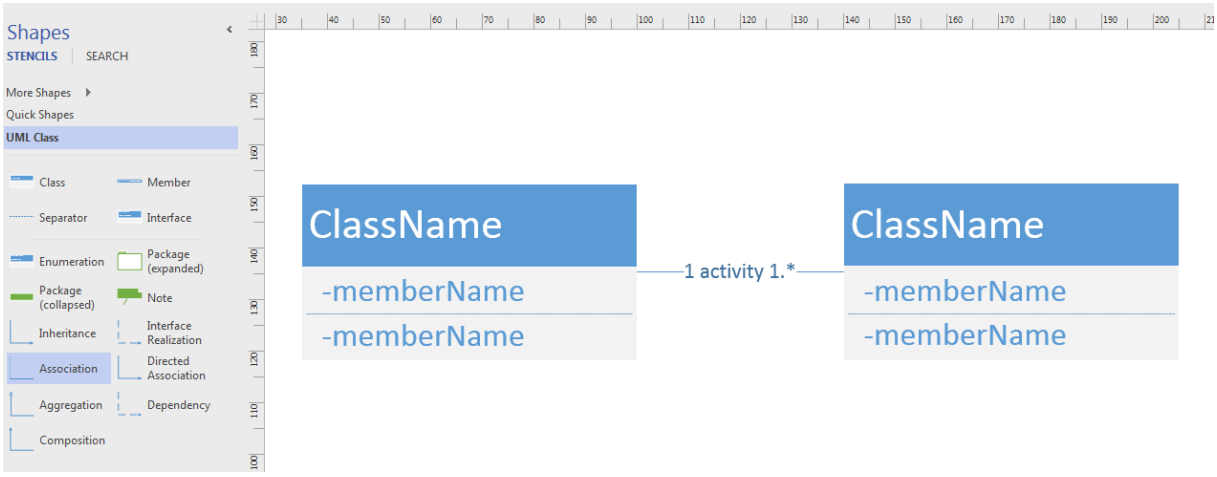


Şekil 5. Kavramsal sınıf diyagramında sınıf ve ilişki tanımı

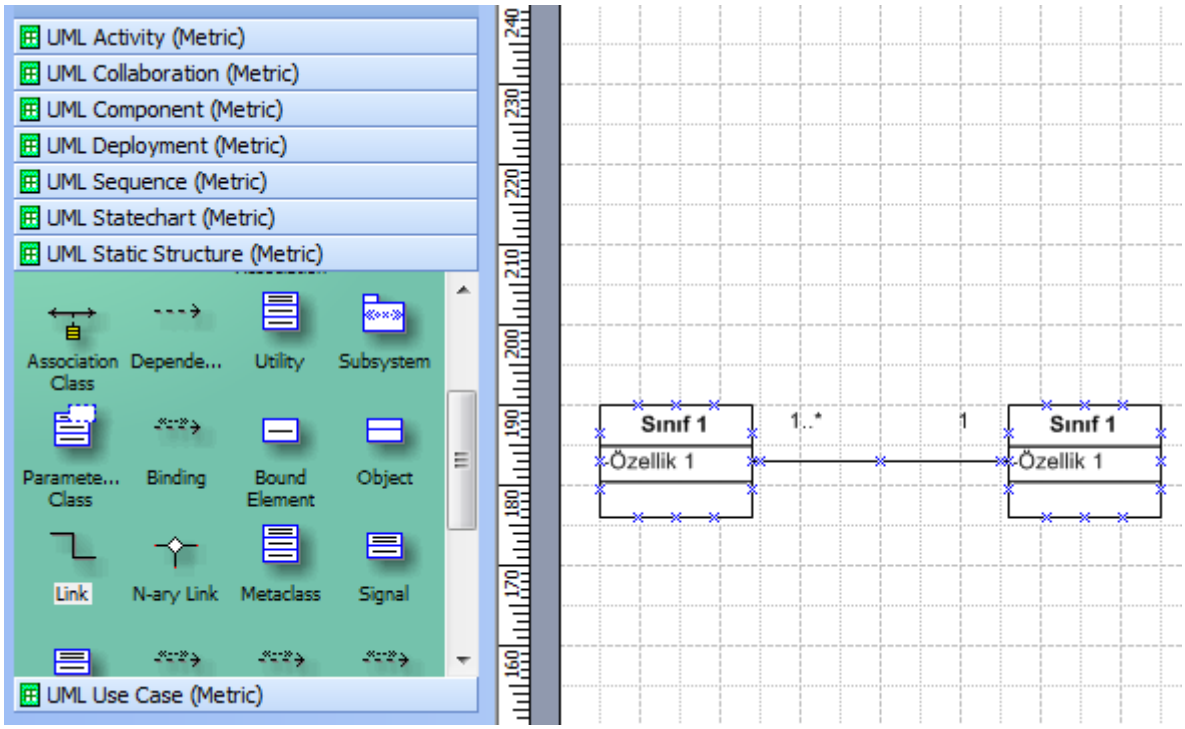
Sınıf Diyagramları Çizimi

Sınıf diyagramlarını MS Visio ile çizebiliriz. Kategoriler içerisinden “Software and Database” kategorisi seçilir ve “UML Class” diyagramı seçilir. “Metric Units” seçilip, “create” butonuna basılır. Soldaki şekilleri sürükle bırak ile diyagramı çizebiliriz. Elemanlara çift tıklayarak isimlendirme yapabiliriz. Sınıflar arası ilişkilendirme "Association" oku ile yapılmaktadır. MS Visio için, oka çift tıklayarak aradaki ilişki türünü okun üzerine yazabiliriz.

MS Visio 2003 te "Software" içerisinde "UML Model Diagram (Metric)" seçilir. Açılan pencerede sol tarafta UML diyagramlarının ismi bulunmaktadır. "UML Static Structure" sekmesinin altındaki elemanlar ile sınıf diyagramını çizebiliriz. "Class" ile sınıf ve nitelik tanımı, "Link" elemanı ile sınıflar arası bağlantı yapılır. Sürükleyip, çizime bıraktığımız "class" elemanına çift tıkladığımızda açılan pencereden sınıf isimlendirmeleri, açılan pencerede "attributes" seçeneğinden "new" butonuna basarak sınıf özelliği tanımlaması yapılır. "Link" elemanı ile sınıfları bağladıktan sonra, "link" elemanının üstüne çift tıkladığımızda açılan pencerede "End1", "End2" yazan kısımları değiştirip, ilişki türü ("1","1..*" gibi) belirtmesi yapabiliriz.

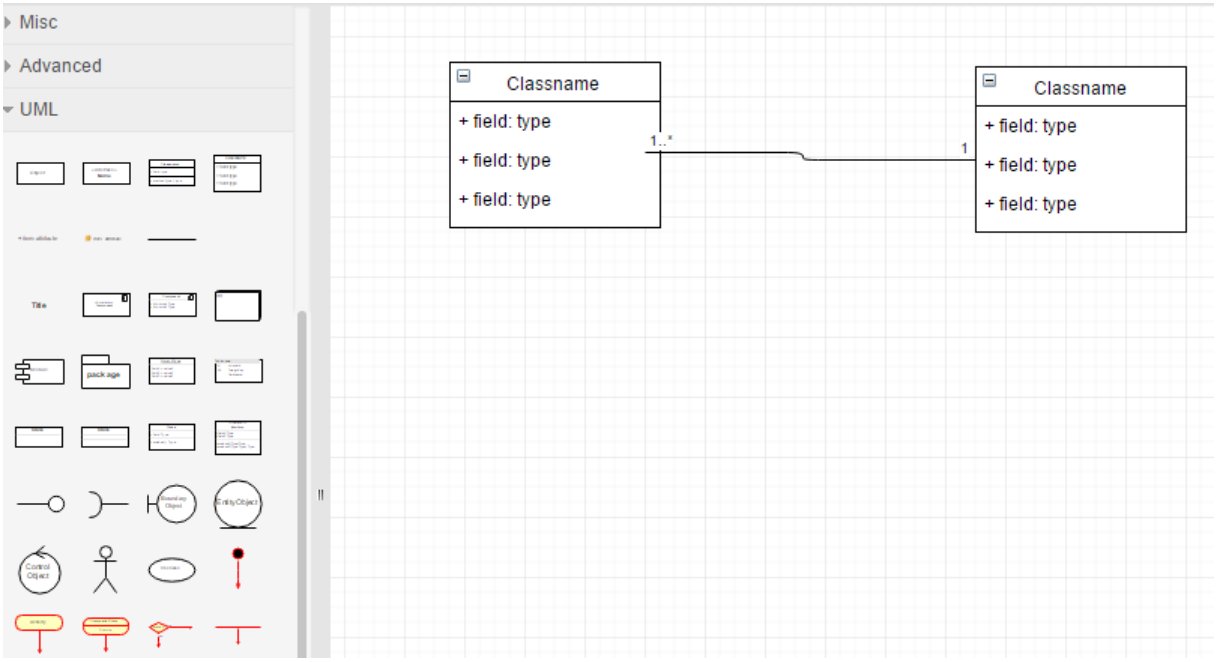


Şekil 6.1. MS Visio 2016 sınıf diyagramı çizimi



Şekil 6.2. MS Visio 2003 ile Sınıf diyagramı çizimi

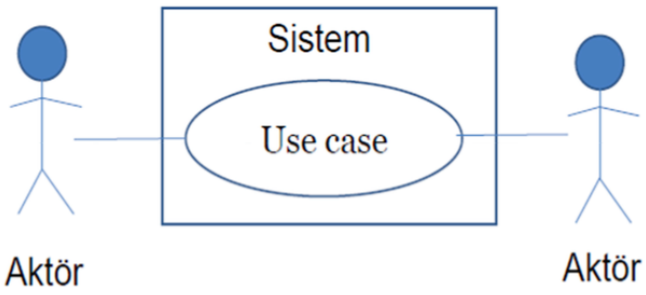
"www.draw.io" adresindeki çizim araçları ile de UML diyagram çizebiliriz. UML sekmesi altında "Class" veya Class 2" elemanı ile sınıf ve nitelik tanımlaması yapılmaktadır. "Association 1" elemanı ile sınıflar birbirine bağlanıp, ilişki türleri belirtilebilmektedir.



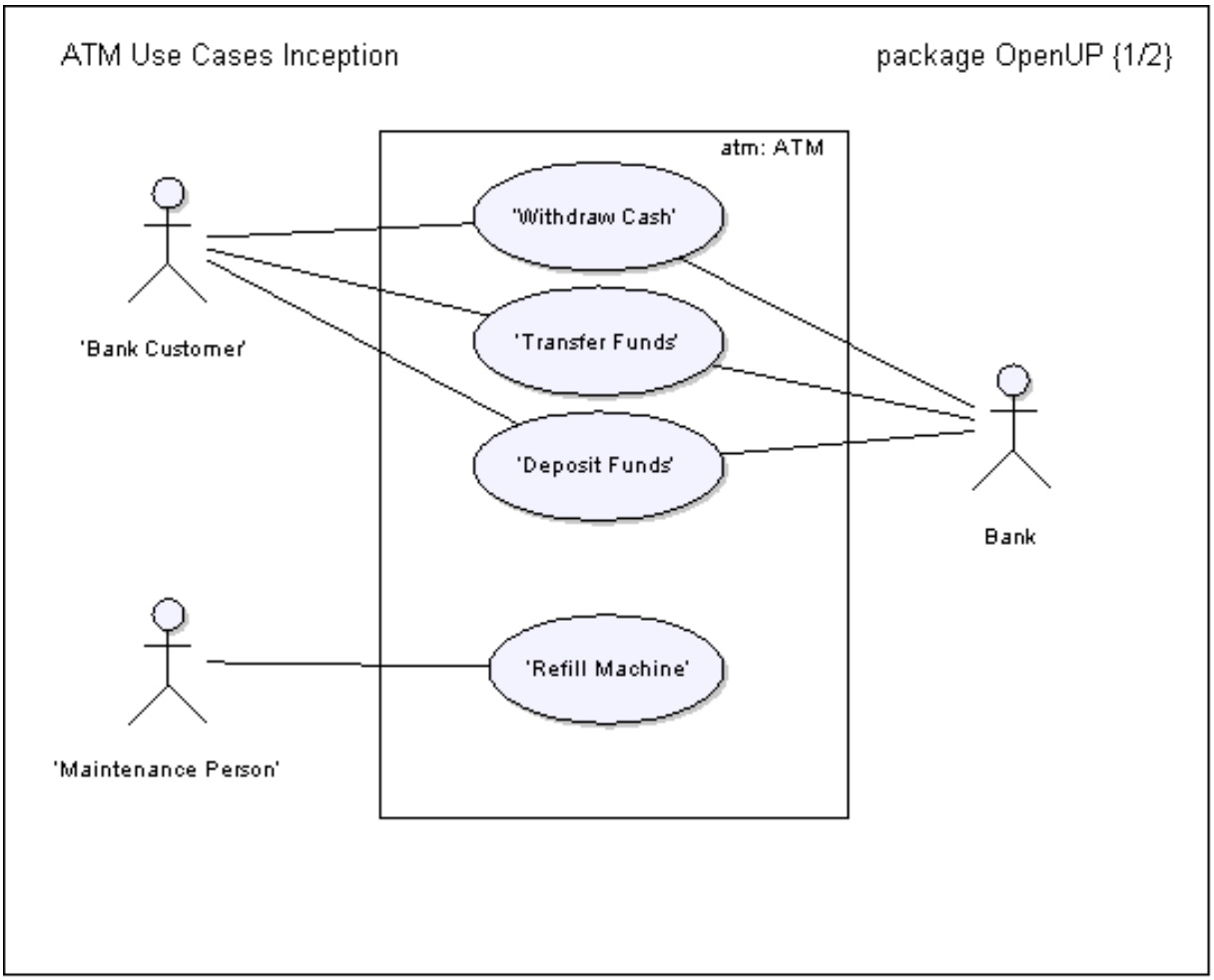
Şekil 6.3. www.draw.io ile UML Class Diyagram Çizimi

Use Case Diagram

Kullanım senaryosu diyagramı (use case diagram), sistem içerisindeki aktörleri ve aktörlerin etkileşimde olduğu olayları gösteren diyagramdır. Kullanım senaryosu diyagramı bir davranış diyagramıdır ve sistemdeki aktörlerden beklenen davranışı modeller. En basit hali ile şekilde gösterildiği üzere aktörler ve davranışlar çizilir.



Şekil 7. 1 Kullanım senaryosu temel elemanları

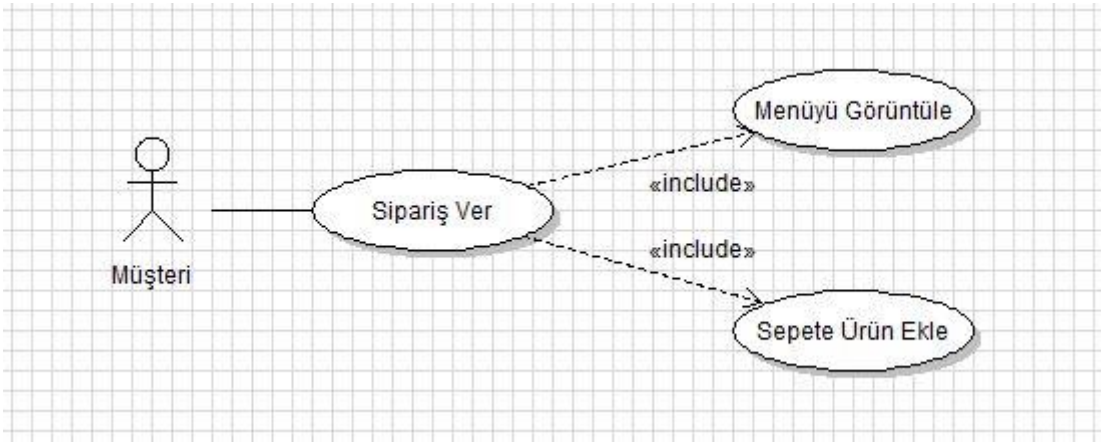


Şekil 7. 2 UML Kullanım diyagramı örneği

Case (senaryo) 'lar arası içerme (include, uses) ve genişletme (extend) ilişkisi bulunabilir.

İçerme (include)

Bazı durumlarda bir use case'in tamamlanabilmesi için, başka use case'lerin işin içerisine katılması zorunludur. Yani söz konusu use case, bir takım başka use case'ler olmaksızın tamamlanamaz. Örnek vermek gerekirse, müşteri, yemeksepeti.com sitesi üzerinden sipariş verebilmek için önce sipariş vermek istediği restoranı seçip, menüyü görüntülemeli ve daha sonra sipariş etmek istediği ürünü / ürünleri alışveriş sepetine eklemelidir. Yani, "Sipariş Ver" isimli use case'in tamamlanabilmesi, "Menüyü Görüntüle" ve "Sepete Ürün Ekle" isimli use case'lerin çalıştırılmasına bağlıdır. Include olarak adlandırdığımız bu ilişki türü, use case diyagramlarında aşağıdaki resimde aktarıldığı şekilde ifade edilmektedir. Burada okların yönü, hangi use case'in tamamlanabilmek için diğer hangi use case'lere bağımlı olduğunu gösterir.



Genişletme (extend)

Use case'ler arasındaki diğer bir ilişki türü extend olarak adlandırdığımız ilişki türüdür. Bu ilişki türü, iki use case arasında opsiyonel olarak var olabilecek bir durumu ifade etmek için kullanılır. Biz bu iki use case'i "base use case" ve "extending use case" olarak adlandıracağız.

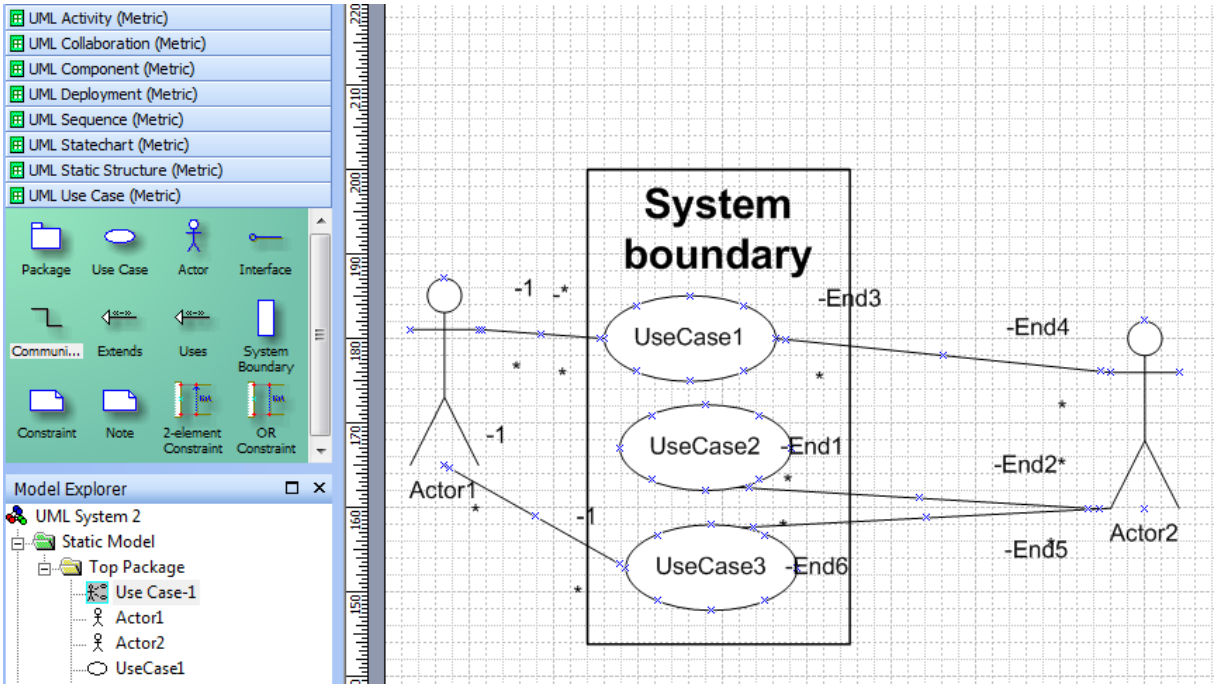
Bu ilişki, bir use case'in kapsamının başka bir use case tarafından genişletilebilmesinin olası olduğu durumlar için geçerlidir. Elimizde bir use case ve bu use case'in kapsamını oluşturan bir dizi adım olduğunu varsayarsak; sürecin bazı noktalarında, başka bir use case'in kapsamını oluşturan adımların kullanılması suretiyle, use case'in adımlarına yeni adımlar eklenmesi ve kapsamının genişletilmesi mümkün olabilir.

Bu durumu bir örnekle anlatalım. Farzedin ki, ATM kullanarak kredi borcunuzu ödüyorsunuz. Bu use case'in ismi "Kredi Ödemesi Yap" olsun. Ödemenizi yaptıktan sonra ATM size makbuz isteyip istemediğinizi soracaktır. İşlemi tamamlamak için makbuz almak gibi bir zorunluluğunuz yoktur ama ATM size böyle bir seçenek sunar. Bu örnekte, "Kredi Ödemesi Yap" base use case'imiz olurken; "Makbuz Yazdır" extending use case'imiz olacaktır ve iki use case arasındaki ilişki bir extend ilişkisidir.

Diyagram üzerindeki okun yönü bize use case'ler arasındaki bağımlılığın yönünü gösterir. Extending use case'in, base use case'e bağlı olduğu unutulmamalıdır. Yani bir extending use case, base use case olmadan tek başına bir anlam ifade etmez. Kredi borcu yatırmadan, borcunuzu yatırdığınıza ilişkin makbuz alamazsınız. Dolayısı ile extend okunun yönü türetilen senaryo'dan ana senaryoya doğrudur.

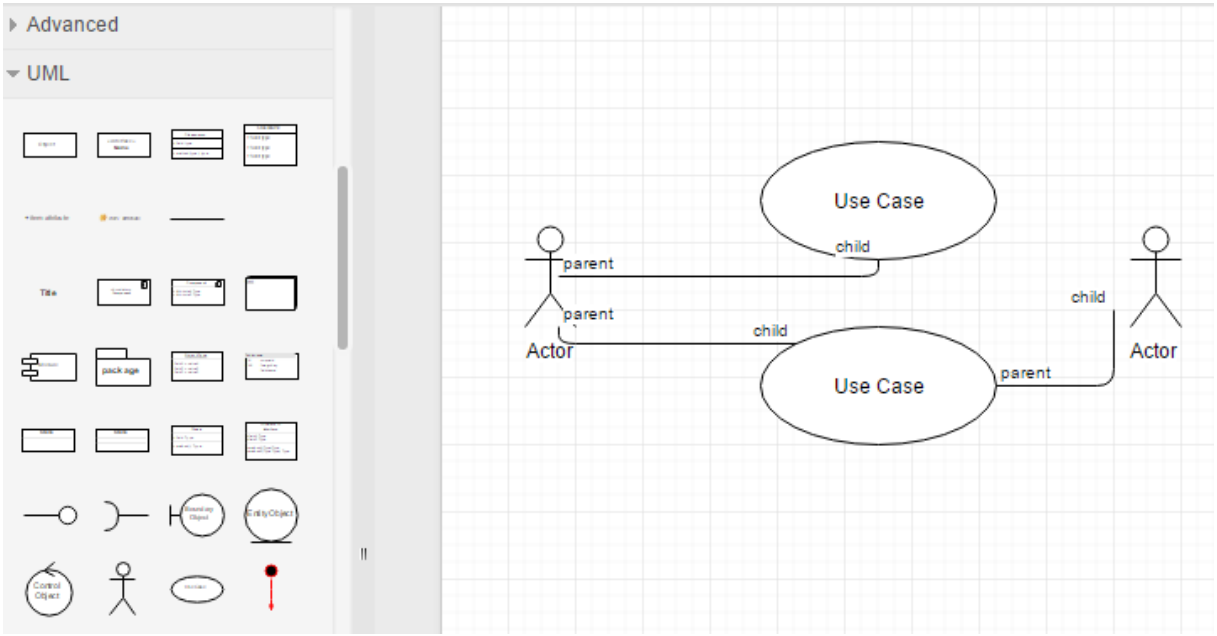
Kullanım senaryosu diyagramı çizimi

MS Visio 2003 için "Software" kategorisi altında "UML Model Diagram(metric)" seçilir. Açılan projede soldaki şekiller kısmında "UML Use case(metric)" seçeneğinin altındaki öğeler ile diyagramı çizebiliriz. Elemanların üstüne çift tıklayıp, açılan pencereden isimlendirme yapılabilmektedir. "Aktor" elemanı ile aktörler; "system boundary" ile davranışları kapsayan sistem; "use case" elemanı" ile kullanım senaryoları belirtilir. "connection" oku ile kullanıcıları, kullanım senaryolarına bağlayabiliriz. Oka çift tıklayıp "end1,end2" ile gösterilen değerleri değiştirerek ilişki türü tanımlanabilir.



www.draw.io adresinde UML klasörü altındaki kullanım diyagramına ait olan elemanlar ile kullanım senaryosu çizilmektedir; ancak sistem sınırı belirten eleman bulunmamaktadır. Association ile aktörler senaryolara bağlanır, uses(include) ve extend ile senaryolar arası içerme/genişletme belirtilebilir.

Şekil 7. 3 MS Visio 2003 kullanım senaryosu diyagramı çizimi



Şekil 7. 4 Draw.io adresinden kullanım diyagramı çizimi

ARDIŞIL DİYAGRAM – YAPI DİYAGRAMI

Sistem Analizi ve Tasarımı Dersi

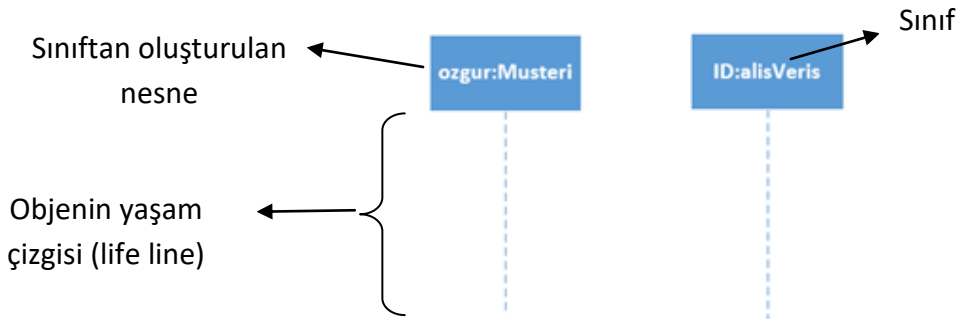
İçindekiler

Ardışıl Diyagram Nedir ve Neden Kullanılır

Ardışıl diyagram (Sequence diagram) nesnelerin birbirleriyle etkileşimini ve etkileşimin sırasını gösteren bir UML diyagramıdır. Literatürde olay diyagramı diye de geçmektedir (event diagram). Ardışıl diyagramlar nesneler (sınıflar/kavramlar) arasındaki mesaj, aksiyon ve olay akışını detaylı olarak kullanıcıya aktarmakta kullanılır. Nesneler arasındaki etkileşimler ardışıl diyagramda ardışıl zamanlarla gösterilerek ifade edilir. Sınıf diyagramından farklı olarak ardışıl diyagramlar nesneler arasındaki etkileşimi zaman sıralamasına göre kullanıcıya sunar. Birbirleriyle etkileşim halindeki bilgi/olay akışının kullanıcıya sunulmasıyla nesneler arasındaki etkileşim (mesajlaşma) detaylıca sunulur. Mesajlar akış diyagramında oluşturuldukları sıraya göre yukarıdan aşağıya doğru çizilir. Ardışıl diyagramın problemi karmaşık sistemlerde nesneler arası mesajlaşma yoğunluğu fazla olacağı için tüm bu mesajlaşmanın görselleştirilmesi zaman alan ve dokümantasyon olarak çok yer kaplayan bir iştir.

Ardışıl Diyagram Elemanları

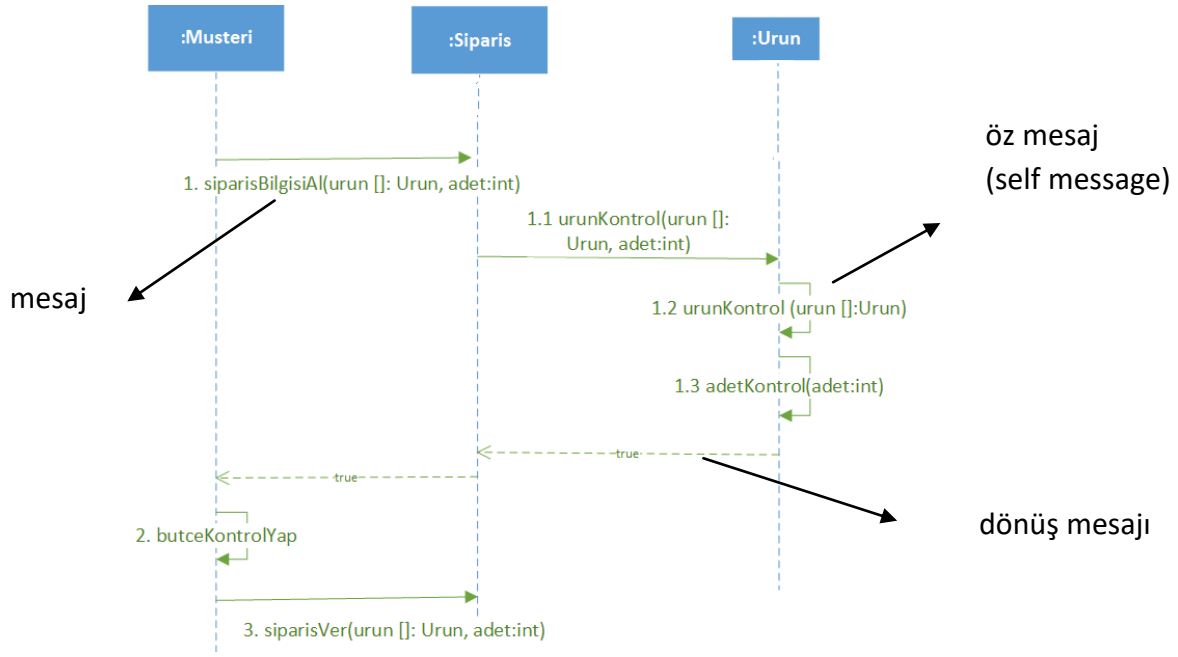
- Katılan (Paydaş veya İştirakçi): Bir nesne veya varlıktır. Şekil 1'de örnek verilmiştir.



Şekil 1. Nesne ve yaşam çizgisi

- Eksenler: Dikey eksen hangi paydaşların etkin olduğunu gösterirken, yatay eksen zamanı göstermektedir.

- Mesaj: Nesneler arasındaki iletişimi gösterir. Şekil 2'de farklı mesaj tipleri verilmiştir.

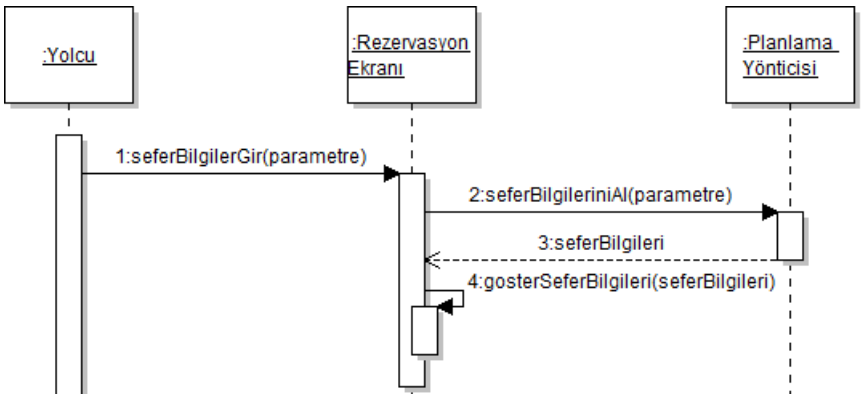


Şekil 2. Mesaj, öz mesaj, dönüş mesajı ve eksenler

Örnek Senaryo :

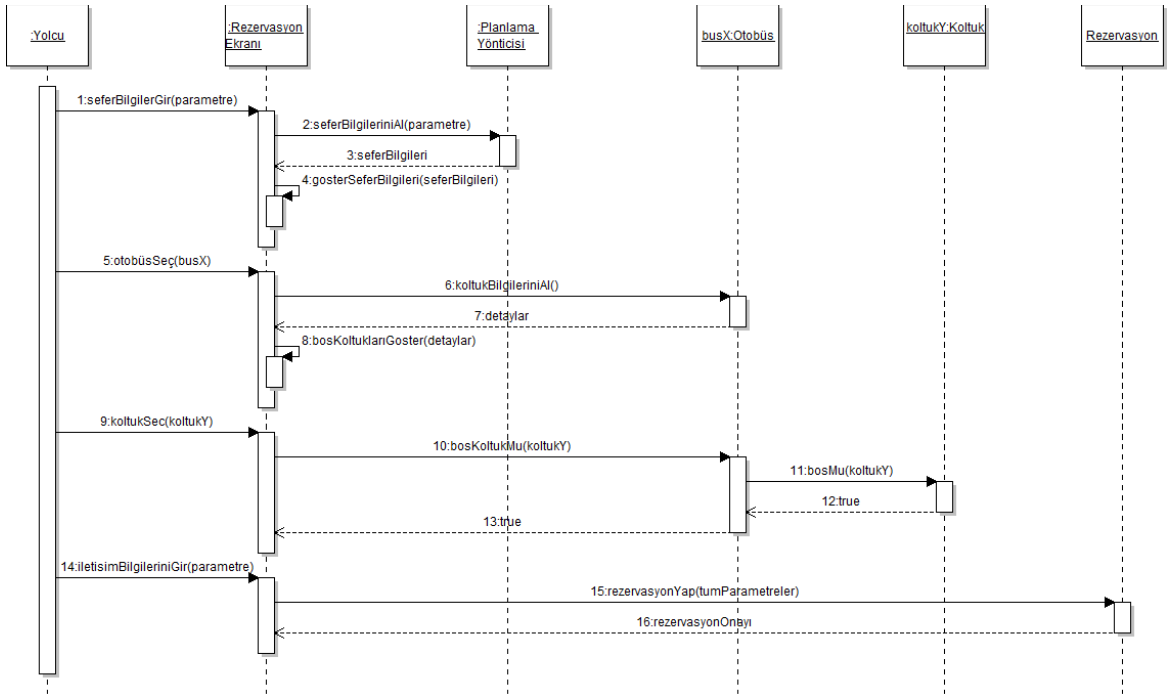
Bir yolcunun yolculuk etmek istediği seferi, istediği seferle ilgili otobüsü ve ilgili otobüsteki koltuğu seçmesini sağlayan ve ilgili bilgilerin rezervasyon bilgisi olarak kaydedildiği bir sistem tasarlanmak istenmektedir. Senaryonun ana akışı gösteren adımlar aşağıda belirtilmiştir.

1. Yolcu seyahat yapmak isteyeceği seyahat bilgilerini girer ve bu bilgilere ait otobüs seferleri sistem tarafından gösterilir.
2. Yolcu bir otobüs seçer.
3. Yolcu seçilen otobüste bir koltuk seçer ve sistem koltuğun uygunluğunu kontrol eder.
4. Yolcu seçimlerini yaptıktan sonra rezervasyon bilgilerini girer ve sistem kaydı gerçeklenir.



Şekil 3. Ardışıl diyagram adımı

Şekil 3'e göre nesne (sınıf) olan yolcu sisteme sefer bilgileri ara yüz olan ekrandan girerek objeler arası etkileşimi başlatmaktadır. Senaryoda aynı saatte aynı kalkış ve istikamete birden fazla otobüs olacağı öngörülerek senaryonun ana akışında varlık olarak bulunmamasına rağmen "PlanlamaYöneticisi" adında yeni bir sınıf tanımlanmıştır. Şekil 3'te sefer bilgileri rezervasyon ekranına yansıtılır ve bu işlem için "RezervasyonEkranı" sınıfı öz metodunu kullanır.



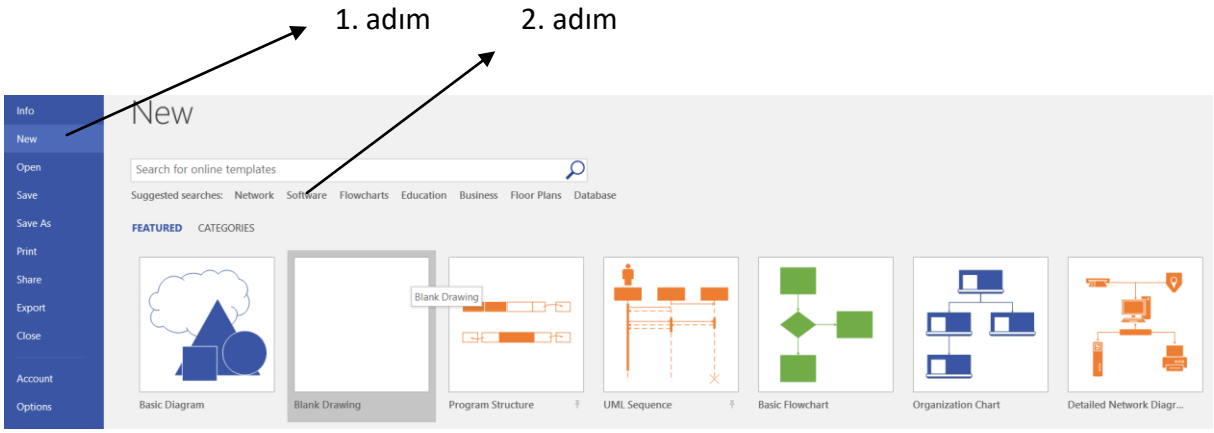
Şekil 4. Senaryonun bütün ardışıl diyagramı

Kullanıcı sefer bilgisi gördükten sonra seçtiği ve parametre olarak gönderdiği (*busX*) koltuk bilgilerini alır ve koltuk seçiminde bulunur. Bu işlemleri tamamladıktan sonra iletişim bilgileri girerek rezervasyon işlemini tamamlar. Şekil 4'te tüm akış sırası mesajlar ile detaylıca gösterilmiştir.

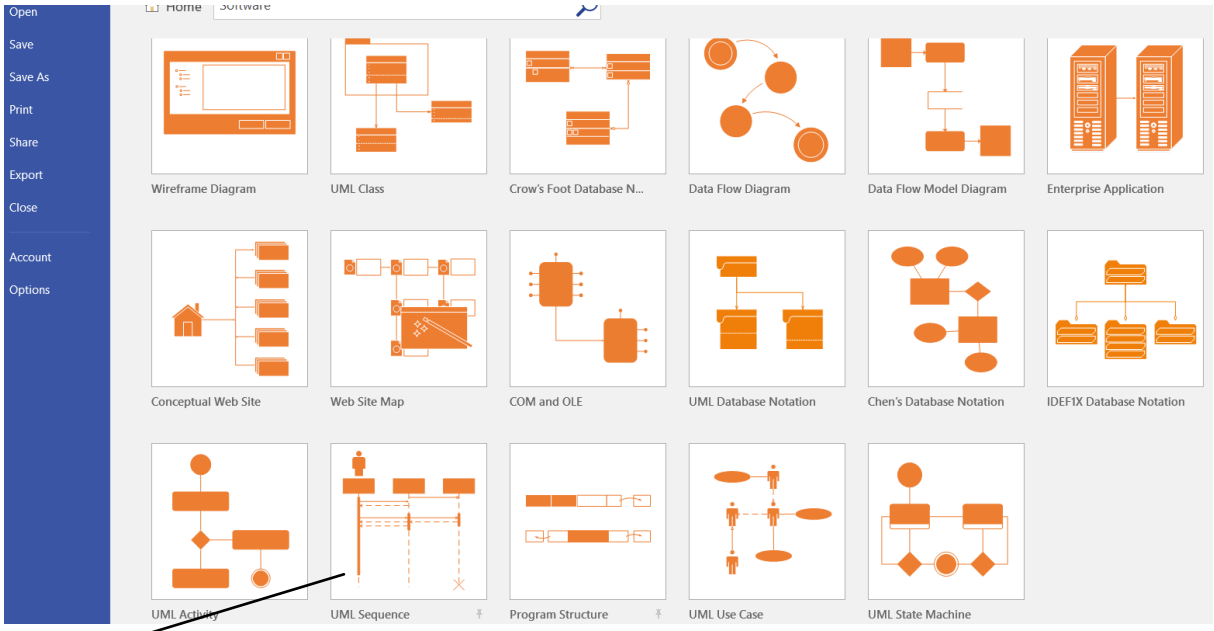
MS Visio ile Ardışıl Diyagram Çizimi

MS Visio 2016 ile Ardışıl Diyagram çizimi için:

New ---> Software ----> UML Sequence---> Create

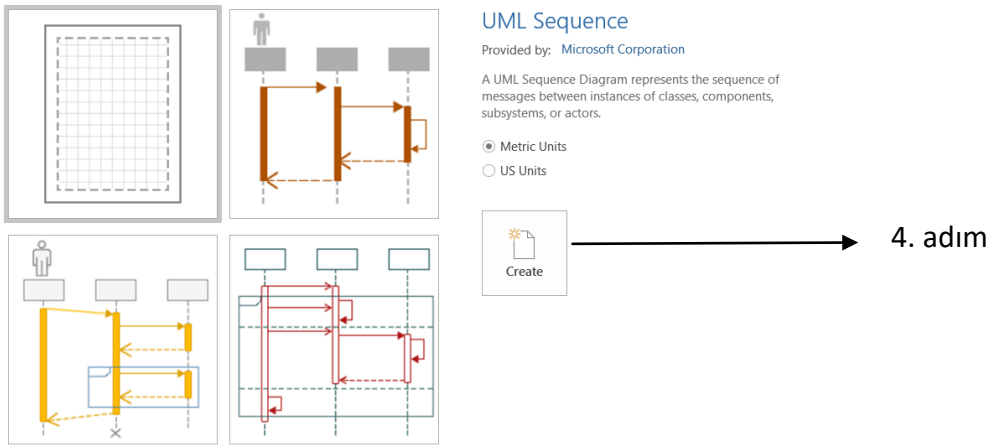


Şekil 5. MS Visio 2016 ile yeni ardışıl diyagram oluşturma- Adım 1



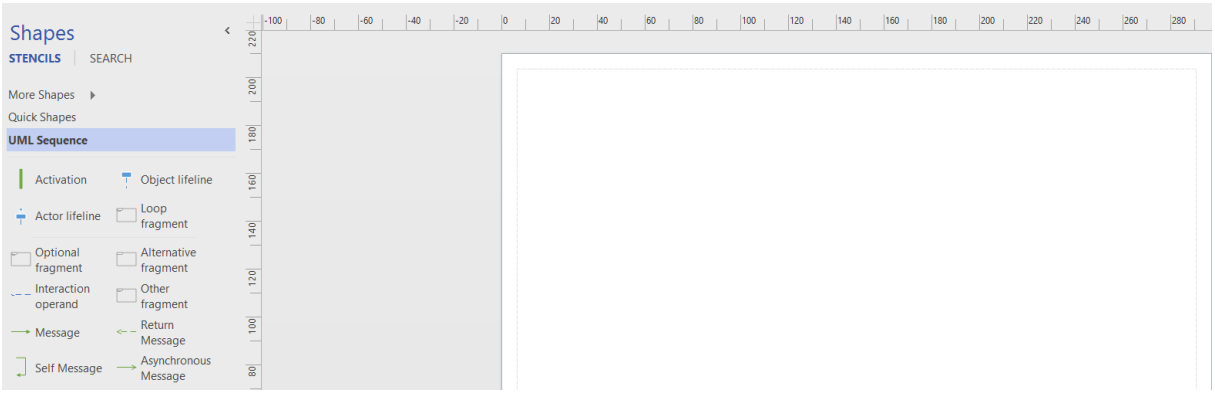
Şekil 6. MS Visio 2016 ile yeni ardışıl diyagram oluşturma - Adım 2

3. adım



Şekil 7. MS Visio 2016 ile yeni ardışıl diyagram oluşturma - Adım 3

Şekil 5-7'de yer alan alanları sırasıyla kullanıcının tıklamasıyla yeni bir Ardışıl Diyagram dosyası oluşturulmuş olacaktır. Sırasıyla Şekil 5'te *New --->Software* ile seçerek yazılım alanındaki diyagramları görmekte olan kullanıcı Şekil 6'daki gibi *UML Sequence* ile ardışıl diyagramı seçer. Şekil 7'deki son işlem olan *Create* adımıyla yeni bir ardışıl diyagram oluşturur.

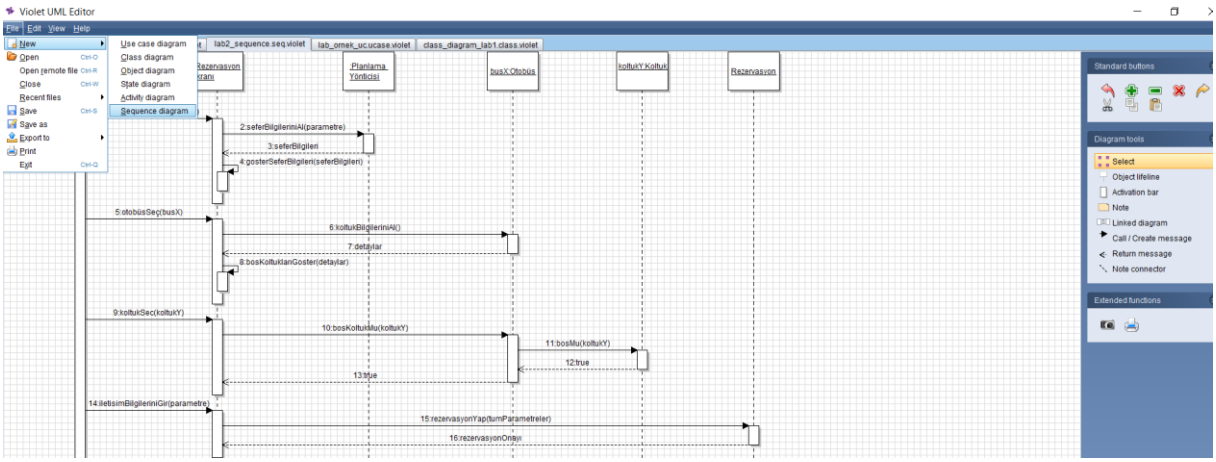


Şekil 8. MS Visio 2016 ile ardışıl diyagram oluşturma

Şekil 8'de "Object Lifeline" sürükle bırakla aktörler (paydaş), "Message" kontrolü sürükle bırakla mesajlar, "Self Message" sürükle bırakla öz mesajlar, "Return Message" kullanarak ise dönüş mesajları nesneler için tanımlanabilmektedir.

Violet UML ile Ardışıl Diyagram Çizimi

Violet UML programı açıldıktan sonra kullanıcının yeni bir ardışıl diyagram oluşturmak için yapması gerekenler Şekil 9'da verilmiştir. Kullanıcı New---> Sequence diagram menüsünü kullanarak yeni bir ardışıl diyagram dosyası oluşturabilmektedir.



Şekil 9. Violet UML ile ardışıl diyagram oluşturma

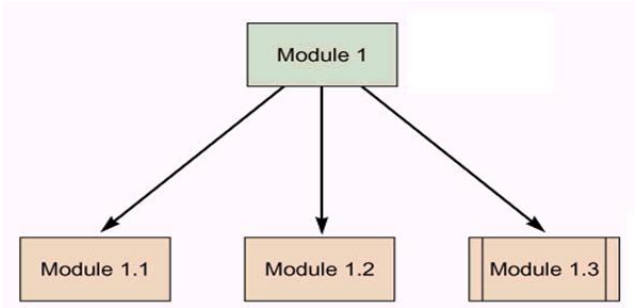
Violet UML kullanarak Ardışıl Diyagram çizilirken mesaj oluşturmak için "Call/Create Message", obje ve yaşam çizgisi oluşturmak için "Object Lifeline" ve dönüş mesajı için "Return Message" kontrolleri kullanılmaktadır.

Yapı Diyagram Nedir ve Neden Kullanılır

Yapı diyagramı sistemdeki (yazılımdaki) yönetilebilen seviyedeki her bir parçanın (modülün) fonksiyonunu görüntülemek için kullanılan diyagramdır. Modüllerin her biri spesifik bir görevi (fonksiyonu) gerçekleştirmektedir. Yapısal programlamada (structured programming) kullanılan bu diyagramlar program modülleri bir ağaç yapısı şeklinde gösterilir. Her bir modül bir dikdörtgen şeklinde gösterilmekle birlikte modülün adı dikdörtgen içerisinde yazmaktadır. Ağaç yapısı modüller arası ilişkileri ve akışı göstermektedir.

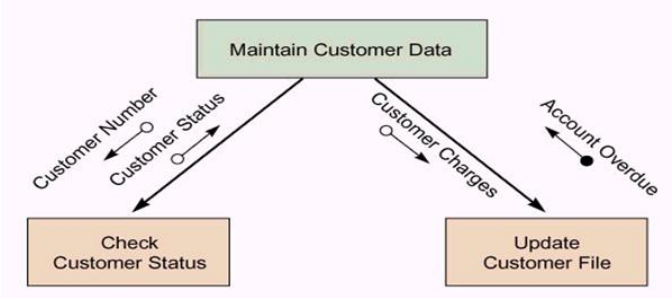
Yapı Diyagram Elemanları

- Modül (program veya alt program): Şekil 10'daki "Module 1" isimli modül alt modüllere ulaştığı için kontrol modülüdür (control module). "Module 1.3" isimli modül birden fazla yerden ve farklı kontrol modülleriyle ilişkili olabileceği için bu tarz modüller kütüphane modülü (library module) olarak adlandırılmışlardır.



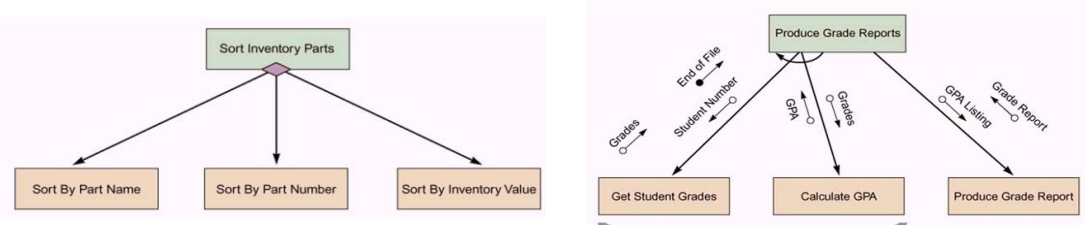
Şekil 10. Yapı diyagramında modül kontrolü

- Veri (data): Şekil 11'deki içi boş olan oklar akışları gösterir. Okun yönü aynı zamanda akışın da yönüdür. İçi dolu olan oklar ise kontrol akışıdır ve işlemin tamamlandığını ve/veya işlem onayını gösterir.



Şekil 11. Yapı diyagramında veri kontrolü

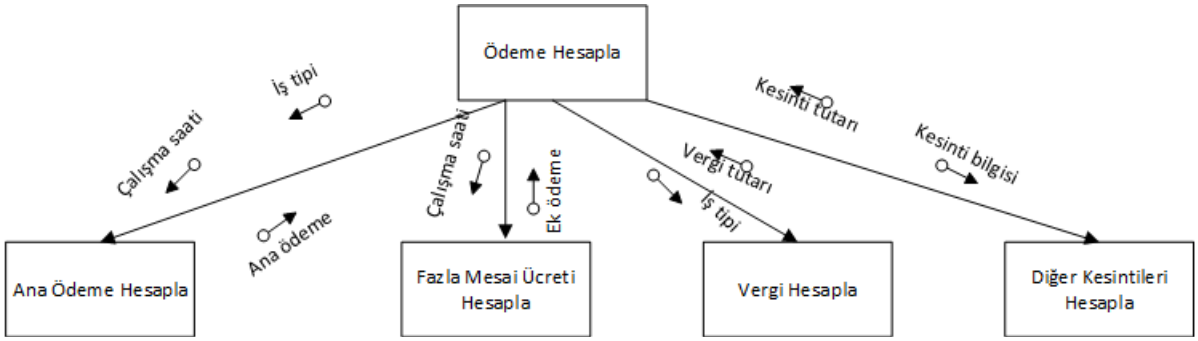
- Döngü ve Durum Kontrolü: Şekil 12'deki gibi yapı diyagramında durum kontrolü elmas şeklinde, döngü ise kavisli ok şeklinde gösterilmektedir.



Şekil 12. Yapı diyagramında döngü ve durum kontrolü

Şekil 12'de envanter parçalarının sıralamasındaki farklı seçenekler durum kontrolüyle gösterilmiştir. Durum kontrolü kontrol modülünün alt modülü aktive ederken farklı seçenekleri olduğunu göstermektedir. Şekil 8 aynı zamanda döngü modülünün kullanımını kullanıcıya açıklamaktadır. Öğrencinin numaraları alındıktan sonra öğrencinin tüm notları elde ediliyor ve her bir öğrenci için genel not ortalaması hesaplanmaktadır. Bu tüm öğrenciler için gerçekleştirildiği için döngü kontrolü kullanılmıştır.

Örnek Senaryo 1:



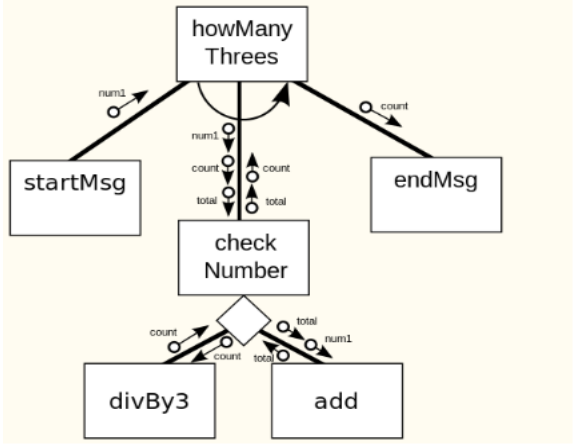
Şekil 13. Yapı diyagramı örnek senaryo

Şekil 13'te "Ödeme Hesapla" kontrol modülünün altında 4 farklı modül bulunmaktadır. "Ödeme Hesapla" ana modülü alt modülleri kullanarak fonksiyonellik kazanmaktadır. Bu alt modüllerin aldıkları verileri ve ürettikleri veriler şekilde gösterilmiştir. Yapısal programlamada alınan veriler fonksiyonlardaki parametrelere, ürettikleri veriler ise dönüş değerine (return value) denk gelmektedir. Buna göre "Ana Ödeme Hesapla" alt modülüne çalışma saati ve iş tipi parametrelerini alarak ana ödeme değerini hesaplamakta ve bu değeri döndürmektedir. 4 alt modülü kullanarak kontrol modülü fonksiyonu yerine getirilmiş olur. Senaryoda yer alan 4 modül birbirinden bağımsız olarak belirtilmiştir. Yapı diyagramlarında iş akışı aynı seviyedeki alt programlar soldan sağa doğru olmakla birlikte alt programlar birbirinden bağımsız olabilmektedir.

Örnek Senaryo 2: Yapısal programlamada kaynak kod kullanarak yapısal diyagramlar elde edilebileceği gibi, yapısal diyagramlardan da taslak kaynak kodlar elde edilebilir. Şekil 14'te bu duruma örnek bir senaryo verilmiştir.

```
sub howManyThrees()
  dim num1, count, total as integer
  num1 = startMsg()
  count = 0
  total = 0
  while num1 > 0 do
     checkNumber(count, total, num1)
     num1 = num1 - 1
  end while
  endMsg(count)
end sub

sub checkNumber(byRef c, byRef t, byVal n)
  If n MOD 3 = 0 Then
     c = divBy3(c)
  Else
     t = add(n, t)
  EndIf
end sub

function divBy3(x)
  return x + 1
end function

function add(n, t)
  return n + t
end function

function startMsg()
  console.writeline("program started, enter your number")
  return console.readline()
end function

sub endMsg(n)
  console.writeline("number of threes : " & n)
end sub
```

Answer :



Şekil 14. Yapı diyagramı- kaynak kod

Şekil 14'teki yapı diyagramı kod ilişkisine göre "howManyThrees" fonksiyonu "startMsg" ile kullanıcıdan alınan değerin içerisinde kaç tane üçün geçtiğini gösteren bir fonksiyondur. Sonunda elde edilen değer "endMsg" fonksiyonu ile yazdırılmaktadır. Buna göre " howManyThrees" fonksiyonundaki alt fonksiyon olan "checkNumber" fonksiyonu ilk değer olarak ilk iki parametreyi sıfır, son parametreyi de içerisinde üç bulacağı sayı olarak alır. Üçüncü parametre değerini bir döngü içerisinde sıfıra eşit olana kadar azaltılır. Bu azaltma işlemlerinde önce kullanıcıdan alınan sayı üçe tam olarak bölünüyorsa ilk parametre değeri (ilk değeri sıfırdı) bir arttırılır ve dönüş değeri olarak belirlenir. Eğer azaltılan sayı üçe bölünmüyorsa ikinci ve üçüncü parametre toplanarak dönüş değeri belirlenir. Döngü sonunda kullanıcıdan elde edilen ilk parametre değeri yani kullanıcıdan alınan sayının

içerisinde kaç defa üçün geçtiğini gösteren değer "endMsg" ile yazdırılarak kontrol fonksiyonu işlevini yerine getirmiş olur.

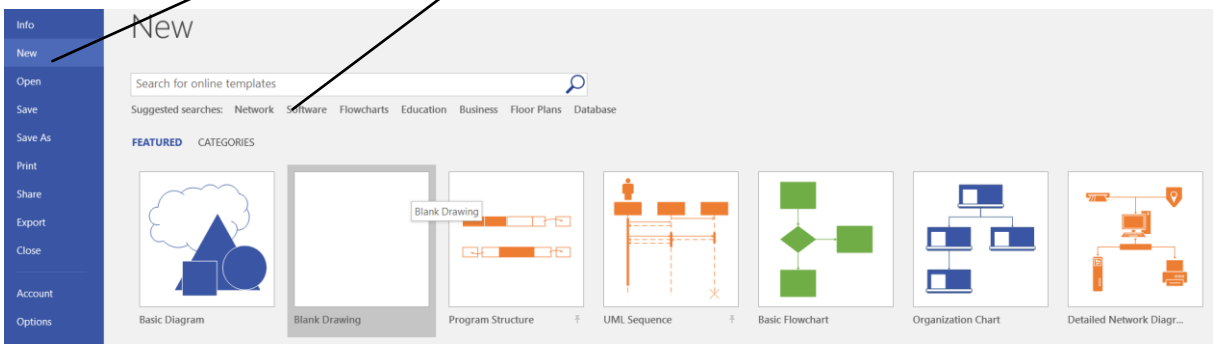
MS Visio ile Yapı Diyagram Çizimi

MS Visio 2016 ile Ardışıl Diyagram çizimi için:

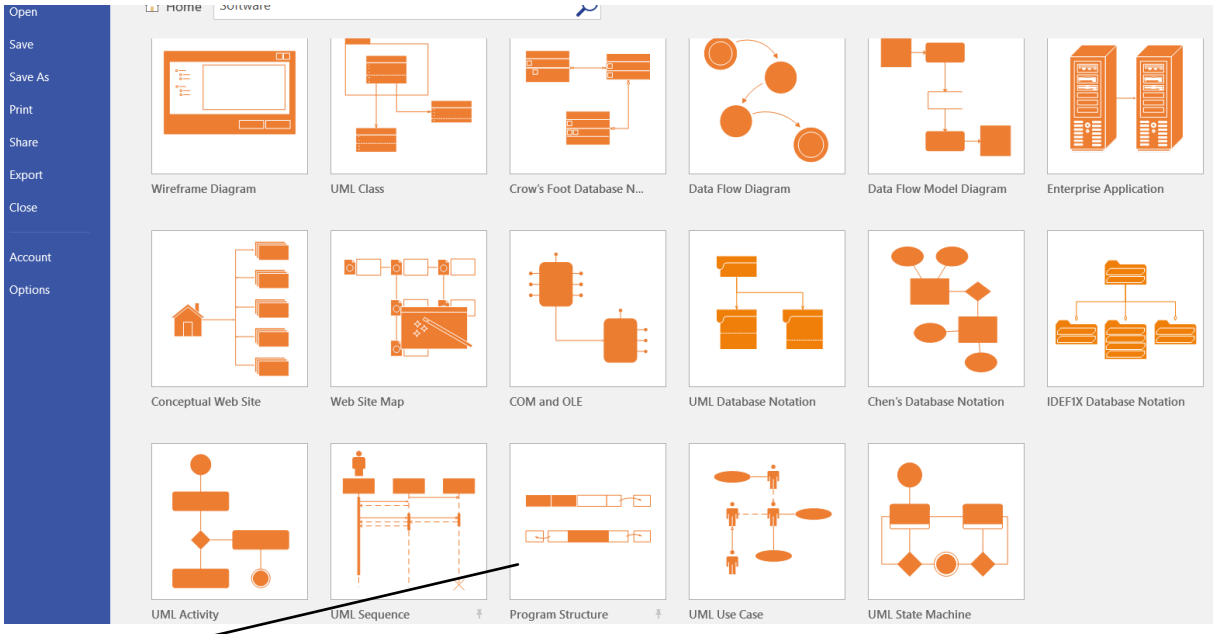
New ---> Software ----> Program Structure---> Create

1. adım

2. adım

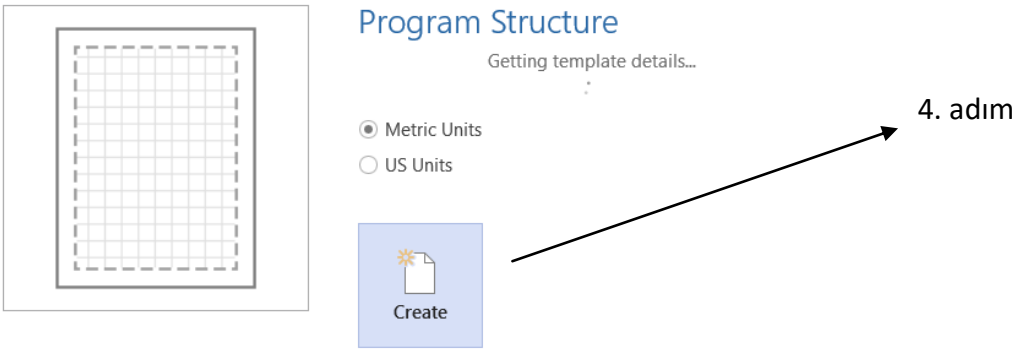


Şekil 15. MS Visio 2016 ile yeni yapı diyagram oluşturma- Adım 1



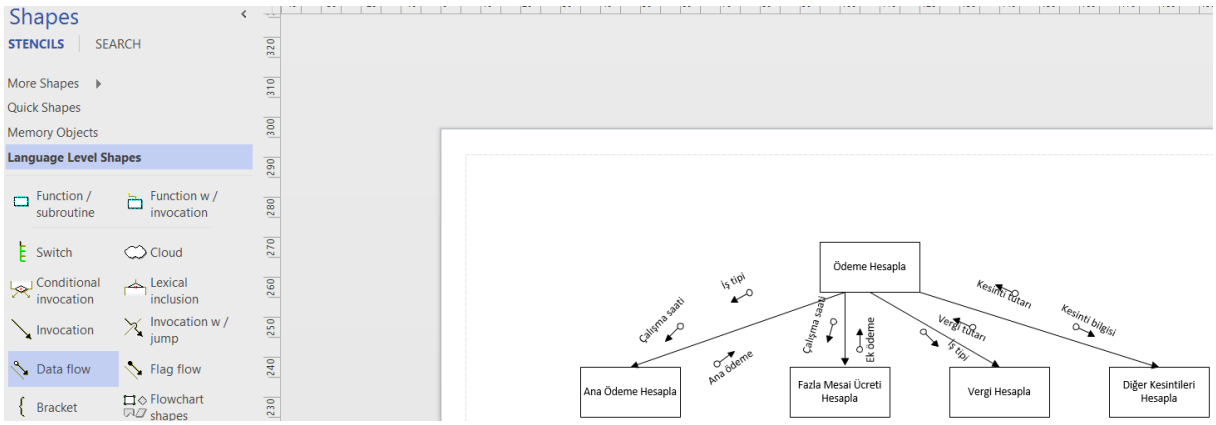
Şekil 16. MS Visio 2016 ile yeni yapı diyagram oluşturma- Adım 2

3. adım



Şekil 17. MS Visio 2016 ile yeni yapı diyagram oluşturma- Adım 3

Şekil 15-17'de yer alan alanları sırasıyla kullanıcının tıklamasıyla yeni bir Yapı Diyagram dosyası oluşturulmuş olacaktır. Sırasıyla Şekil 15'te *New --->Software* ile seçerek yazılım alanındaki diyagramları görmekte olan kullanıcı Şekil 16'daki gibi *Program Structure* ile yapı diyagramını seçer. Şekil 17'deki son işlem olan *Create* adımıyla yeni bir yapı diyagramı oluşturur.



Şekil 18. MS Visio 2016'daki yapı diyagramı kontrolleri

Şekil 18' de gösterilmiş kontrollerin detayları aşağıda kısaca özetlenmiştir. Buna göre :

- Function/subroutine: Kontrol modülü
- Function w/ invocation: Alt modül
- Invocation: Bağlantılar
- Data flow: Veri akışı
- Flag flow: Kontrol akışı
- Flowchart Shapes: Döngü ve diğer kontroller
- Conditional Invocation: Durum kontrolü